

السلام عليكم ورحمة اللة وبركاتة انا طالب ادرس في قسم هندسة برمجيات في روسيا جامعة الدون الحكومية الدولية في مدينة روستوف نادانو.

المهندس عبد الماجد الخليدي
طالب دكتوراة في قسم تحليل انظمة معلومات وادارة المعلومات ومعالجتها ونظم السيطرة.

الموهل الذي امتلكة مهندس وماجستير في قسم هندسة برمجيات الحاسوب والانظمة الاتوماتيكية بتقدير امتياز مع مرتبة الشرف .

السيرة الذاتية:يمني محافظة تعز
العمر: 25 سنة

email:alkhulaidi_2006@hotmail.com
abdulmajed1983@yahoo.com
icq : 434-425-244
web site : abdulmajed.8m.net
موقع الجامعة:**http://www.dstu.edu.ru**

الغات وقواعد البيانات و انظمة التشغيل التي اجيدها في البرمجة:

c , c++, delphi ,delphi in linux(kylix), pascal , assembler , basic ,visual basic ,c# , java ,java#,html ,xml,uml, css , frontpage,Macromedia Dreamweaver,java script , php, perl , Macromedia Flash and programming in flash,opengl with c++ for games, delphi for php , programming in unix with c language.

Database: oracle,postgres(postgresql) , ms sql server 2005, mysql ,mysql in linux, Ms access, pradox, Database 1c in russian languag, foxpro.

Operation systems: any windows(xp,98,NT,ME,2000,95,SERVER,VISTA) , linux .

Dynamic data structure

Dynamic data structure
list,queue,stack: تنقسم الى ثلاثة

في هذا الموضوع سوف نتعامل مع الموشرات pointers بترتيب غير ثابت
اي ترتبط بترتيب حركي بحيث ان هذة المؤشرات تغير ترتيب البيانات عن طريق تغير وجهة
المؤشرات فسوف نوضح بعض الامثلة ل ديناميك داتا ستركتر في الباسكال والسي ممكن تكتب
على اي لغة بدك اياة واعلم ان في السي شارب لاتوجد موشرات كما في الباسكال يرمز^
والسي<- .
المؤشرات هي من انواع البيانات البسيطة والمتغيرات المؤشرية تستخدم للاشارة الى متغيرات من
انواع اخرى وهذة تسمى المتغيرات المؤشر عليها .

Stack:(المكدس)
stack(lifo)-last in first out
الاضافة والحذف من النهاية

النهاية 3
5
البداية 11

فاذا ارادنا نضيف العدد 6 فاننا نضيفة بعد العدد 3

النهاية 6
3
5
البداية 11

فاذا ارادنا ان نحذف عنصر فانة سيكون من البداية وهو العدد 6

النهاية 3
5
البداية 11

مثال على الباسكال

```
program Stek2;
uses crt;
const
n=4;
var
x:integer;
a:array[1..n] of integer;
i:integer;
e:integer;
{---------------}
Procedure Add;
var
e:integer;
begin
x:=0;
i:=1;
writeln('STEK free');
repeat
x:=x+1;
writeln('ENTER lement Steka - ',i);
readln(e);
a[i]:=e;
i:=i+1;
until x=n;
```

```
writeln('Stek Full');
end;
{---------------}
Procedure Show;
begin
writeln('++++++++');
for i:=x downto 1 do
begin
writeln(a[i]);
end;
end;
{---------------}
Procedure Delt;
begin
x := x-1;
end;
{---------------}
begin
clrscr;
Add;
Show;
repeat
writeln('delete ');
readkey;
delt;
writeln('position ',x);
show;
if i=x then break;
i:=i+1;
until i=n;
writeln('Stek free');
readln;
readln;
end.
```

G:\pascal\TP\BIN\TPX.EXE

```
STEK free
ENTER lement Steka - 1
11
ENTER lement Steka - 2
5
ENTER lement Steka - 3
3
ENTER lement Steka - 4
6
Stek Full
++++++++
6
3
5
11
delete
position 3
++++++++
3
5
11
delete
```

مثال اخر على السي

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

/* in pascal
type
TMY=^_tmy;
_tmy=record
name :char;
age:integer;
next:TMY;
end;
*/
typedef struct _TMy  TMy;
struct _TMy
{
char* Name;
int   Age;
TMy * Next;
```

```
};

void show( TMy b )
{
printf("Name: %s, age: %d\n", b.Name, b.Age);
}

TMy* Top;

void add_stack(char* s, int a)
{
TMy *d;
d = (TMy *) malloc (sizeof(TMy));
(*d).Name = strdup(s);
(*d).Age = a;
(*d).Next = Top;
Top = d;
}

void show_stack()
{
TMy *tmp = Top;
while (tmp != NULL)
{
show(*tmp);
tmp = (*tmp).Next;
}
}

void main()
{
clrscr();

char st[80];
int age;
for(int i=0; i<3; i++)
```

```
{
flushall();
puts("Enter name: ");
gets(st);
puts("Enter age: ");
scanf("%d", &age);
add_stack(st, age);
}
printf("In base:\n");
show_stack();

}
```

Turbo C++ IDE

```
Enter name:
abdulmajed
Enter age:
25
Enter name:
adilson
Enter age:
26
Enter name:
wadah
Enter age:
25
In base:
Name: wadah, age: 25
Name: adilson, age: 26
Name: abdulmajed, age: 25
```

(الترتيب)Queue(FIFO)

first in first out

الاضافة في البداية والحذف من النهاية

3
5
11

فاذا ارادنا اضافة العنصر 6 فاننا سنضيفة في البداية قبل 11

3
5
11
6

واذا ارادنا الحذف فانة سيتم من النهاية يعني العدد 3

5
11
6

مثال غلى الباسكال

```
uses crt;
type
    tip     =^element;
    element = record
    inf     : integer;
    link    : tip;
          end;
var
```

```
   begQ,endQ,p : tip;
   kon:tip;
   v,i,n : integer;
{-----------------------------------------------------}
procedure cozidat(v1:integer);
var p : tip;
   begin
     new(p);
     p^.inf:=v1;
     p^.link:=nil;

     if begQ = nil then
        begin begQ := p; {dababit tolka adin element}
        endq:=p;{dababit cklka mi xatem}
 end     else
      begin       endQ^ .link:=p;
        endQ:=p; end;
      end;
{---------------------------------------------------}
procedure delete(var v1:integer);
var  m:tip;{p:tip}
   begin
    { v1:=begQ^.inf;
     p:=begQ;
     begQ:=begQ^.link;
     if begQ = nil then
       endQ:= nil;
     dispose(p);   }
     v1:=begQ^.inf;
      m:=begq;
      begq:=begq^.link;

      dispose(m);
   end;
{---------------------------------------------------}
var
```

```
    option : byte;
    key : integer;
    v1: integer;
Begin
  clrscr;
     begQ:=nil;
     endQ:=nil;
write('Enter n = ');
{   readln(n);
 }
     for i:=1 to n do
     cozidat(i);



repeat
     writeln('N-',1,' Enter more element:');
     writeln('N-',2,' delete one element from the end of querre:');
     writeln('N-',3,' show querre:');
     writeln('N-',0,' exit :');

     writeln('Enter a choice:');
     readln(option);

    case option of
    1: begin
        writeln('Enter more element:');
        readln(key);
        cozidat(key);
      end;
    2:  begin
         write('Delete element in queue : ');
         Delete(v1);writeln('that is element : ',v1);
      end;
     3:  begin
        kon:=begQ;
```

```
          writeln('elements in queue : ');
          while kon<>nil do
            begin
              writeln(kon^.inf,'  ');
              kon:=kon^.link;
            end;
          writeln;
        end;
     0: exit;

    end; { of case }
until false;
    readkey;
End.
```

G:\pascal\TP\BIN\TPX.EXE

```
N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
1
Enter more element:
11
N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
3
elements in queue :
3
5
11

N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
```

```
G:\pascal\TP\BIN\TPX.EXE
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
1
Enter more element:
6
N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
3
elements in queue :
3
5
11
6

N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
```

```
G:\pascal\TP\BIN\TPX.EXE

N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
2
Delete element in queue : that is element : 3
N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
3
elements in queue :
5
11
6

N-1 Enter more element:
N-2 delete one element from the end of querre:
N-3 show querre:
N-0 exit :
Enter a choice:
```

مثال اخر على السي

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>

typedef struct _ozer  OZER;
struct  _ozer
{
char*name;
int age;
OZER * next;
};


OZER *BegPtr, *EndPtr;

void add(char* name, int age)
{
OZER *tmp;
tmp = (OZER*) malloc (sizeof(OZER));
tmp->name = strdup(name);
tmp->age  = age;
tmp->next = NULL;

if (BegPtr == NULL) {
// queue is empty
BegPtr = EndPtr = tmp;
} else {
// queue is not empty
EndPtr->next = tmp;
EndPtr = tmp;
}
}
```

```
void show()
{
OZER *tmp;
tmp = BegPtr;
while (tmp != NULL) {
printf("Element:\n  Name:='%s'\n  Age:='%d'\n", tmp->name, tmp->age);
tmp = tmp->next;
}
}

void main()
{
char name[80];
int age;
int i;
for (i=0; i<3; i++) {
printf("\nEnter name: ");
flushall();
gets(name);
printf("\nEnter age: ");
scanf("%d", &age);
add(name, age);
}
show();
getch();
}
```

```
Turbo C++ IDE

Enter name: abdulmajed

Enter age: 25

Enter name: adilson

Enter age: 26

Enter name: wadah

Enter age: 25
Element:
   Name:='abdulmajed'
   Age:='25'
Element:
   Name:='adilson'
   Age:='26'
Element:
   Name:='wadah'
   Age:='25'
_
```

List(القوائم)

الاضافة في اي مكان مسموح في البداية اوفي الوسط او في النهاية اختيارية

3
5
11

فاذا اردت اضافة العدد 6 يمكنك اضافتها

كيفما تريد قبل ال 5 او بعدها او قبل ال 11

والحذف كذلك في اي مكان مسموح في البداية اوفي الوسط او في النهاية اختيارية

امثلة في الباسكال

مثال 1

اعمل generator للعدد واضافتة في القائمة فاذا كان هذا العدد موجود في القائمة فاحذفة من القائمة واذا كان غير موجود اضيفة في القائمة.

```
clrscr;
 add(2);
 add(10);
 add(5);
```

```
  add(1);
  add(7);
  add(11);


  writeln('___');
  show;
   addEx(6);    //لاحظ هذه الدالة التي نفحص فيما اذا كان العدد  موجود او لا فنحن هنا
               //     استدعيناها
```

لاحظ هنا سيتم اضافة العدد 6 لانة لا يوجد ضمن القائمة {2,10,5,1,7,11}

لكن اذا كان ;addEx(10) طبعا سيتم حذفة من القائمة لانة يوجد في القائمة

```
  writeln('___');
  show;
```

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
    i : integer;
    next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;

function find(i : integer) : Tr;
var
  pp, pp1 : Tr;
begin
  pp1 := nil;
  pp := begptr;
  while pp <> nil do
  begin
    if pp^.i > i then
```

```
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find := pp1;
end;

procedure add( i : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
    pp1 := find(i);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;
```

```
procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
```

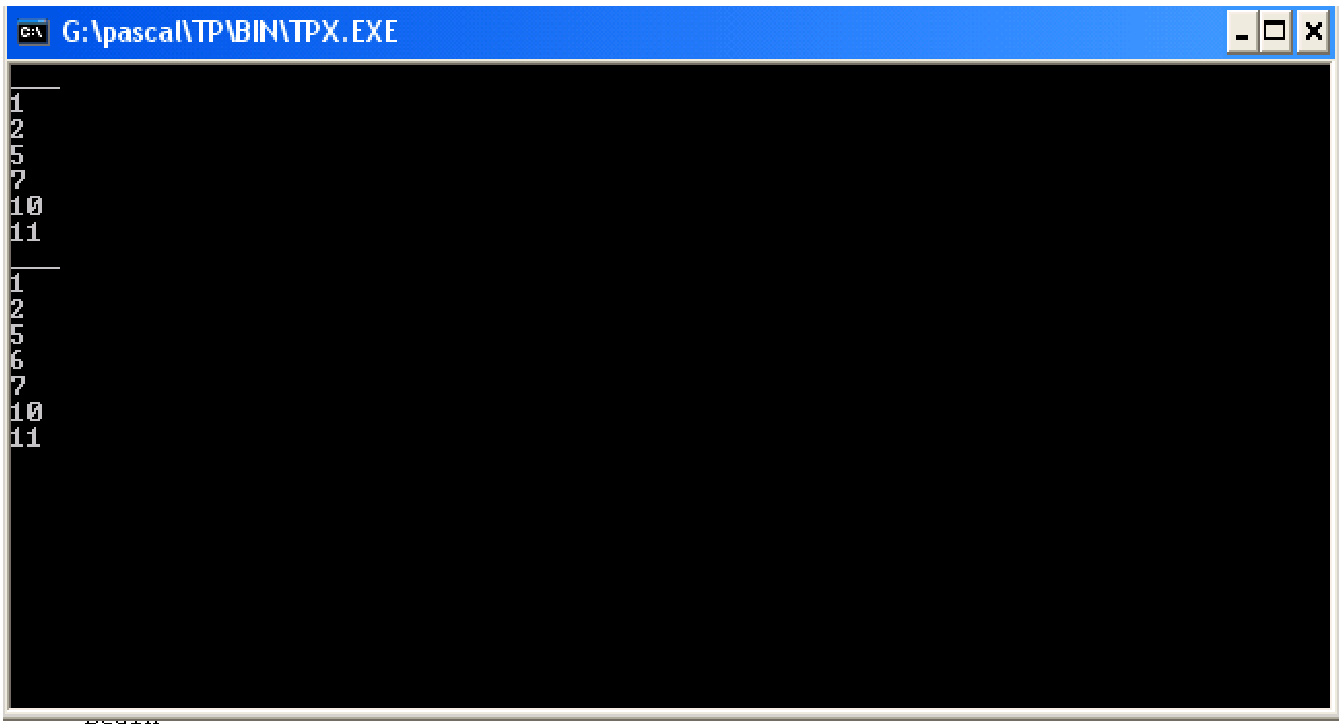
```
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{     del2(pp1);}
  end;
end;
```

```
function find_el(i : integer) : integer;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  find_el := 0;
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pp^.i = i then
    begin
      find_el := ind;
      break;
    end;
    pp1 := pp;
    pp := pp^.next;
  end;
end;

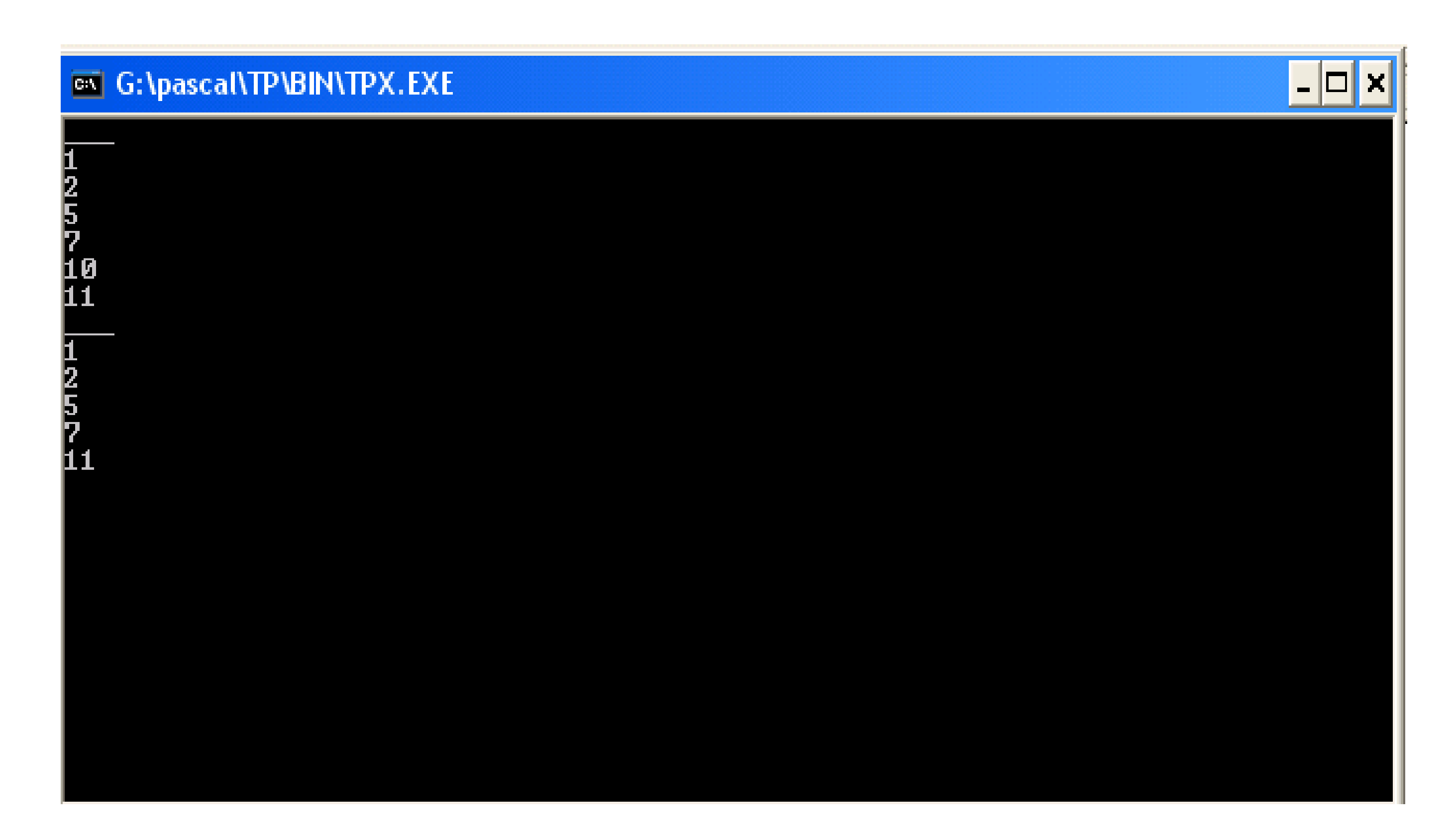
procedure addEx(i : integer);
var
  ind : integer;
begin
  ind := find_el(i);
  if ind = 0 then
    add(i)
  else
    del(ind);
end;

begin
  clrscr;
```

```
  add(2);
  add(10);
  add(5);
  add(1);
  add(7);
  add(11);
  writeln('____');
  show;
  addEx(6);
  writeln('____');
  show;
end.
```



لكن اذا كان addEx(10); طبعا سيتم حذفة من القائمة لانة يوجد في القائمة



المثال الثاني

القائمة معطاة اوجد العنصر المتوسط من عناصر القائمة واوجد اكبر عنصر في الجهة اليسرى للعنصر المتوسط واكبر عنصر في الجهة اليمنى للعنصر المتوسط .

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
    i : integer;
```

```
    next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;

function find(i : integer) : Tr;
var
  pp, pp1 : Tr;
begin
  pp1 := nil;
  pp := begptr;
  while pp <> nil do
  begin
    if pp^.i > i then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find := pp1;
end;

procedure add( i : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
```

```
    pp1 := endptr;
{   pp1 := find(i);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else}
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;
```

```
  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
```

```
        pp2^.next := nil;
        dispose(endptr);
        endptr := pp2;
      end
      else
      begin
        pp2 := prev(pp1);
        pp1 := pp1^.next;
        dispose(pp2^.next);
        pp2^.next := pp1;
      end;
    exit;
  end;
{     del2(pp1);}
 end;
end;

function find_el(i : integer) : integer;
var
 pp, pp1 : Tr;
 ind : integer;
begin
 find_el := 0;
 pp1 := nil;
 pp := begptr;
 ind := 0;
 while pp <> nil do
 begin
   inc(ind);
   if pp^.i = i then
   begin
     find_el := ind;
     break;
   end;
   pp1 := pp;
   pp := pp^.next;
```

```
  end;
end;

procedure addEx(i : integer);
var
  ind : integer;
begin
  ind := find_el(i);
  if ind = 0 then
    add(i)
  else
    del(ind);
end;

procedure middel_and_max;
var
  pp : Tr;
  ind, summ, count, mid : integer;
  dx, i_mid : integer;
  halfcount : integer;
  emax, lmax, rmax : integer;
begin
  summ := 0;
  count := 0;
  ind := 0;
  pp := begptr;
  while pp <> nil do
  begin
    summ := summ + pp^.i;
    inc(count);
    pp := pp^.next;
  end;
  mid := summ div count;

  pp := begptr;
  dx := begptr^.i;
```

```
{  halfcount := count / 2;}
  lmax := 0;
  rmax := 0;
  while pp <> nil do
  begin
    inc(ind);
    if abs(mid - pp^.i) < dx then
    begin
      dx := mid - pp^.i;
      emax := pp^.i;
      i_mid := ind;
    end;
    if (ind < (count/2)) then
    begin
      if lmax < pp^.i then lmax := pp^.i;
    end
    else
    begin
      if rmax < pp^.i then rmax := pp^.i;
    end;
    pp := pp^.next;
  end;
  writeln('Motawaset element:',emax);
  writeln('Max left element:',lmax);
  writeln('Max right element:',rmax);
end;

begin
  clrscr;
  add(2);
  add(10);
  add(5);
  add(1);
  add(7);
  add(11);
  writeln('___');
```

```
  show;
  middel_and_max;
{  writeln('___');
  show;}
end.
```

```
G:\pascal\TP\BIN\TPX.EXE
___
2
10
5
1
7
11
Motawaset element:5
Max left element:10
Max right element:11
_
```

المثال الثالث

القائمة معطاة اوجد اول عنصر اقل من الصفر ومن ثم اعمل عملية ترتيب للعناصر بعد هذا العنصر الذي اقل من الصفر .

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
```

```
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;

function find(i : integer) : Tr;
var
  pp, pp1 : Tr;
begin
  pp1 := nil;
  pp := begptr;
  while pp <> nil do
  begin
    if pp^.i > i then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find := pp1;
end;

function find_place(pl : integer) : Tr;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
```

```
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
{    pp1 := endptr;{
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;
```

```
procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
```

```
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{       del2(pp1);}
  end;
end;
```

```
function find_el(i : integer) : integer;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  find_el := 0;
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pp^.i = i then
    begin
      find_el := ind;
      break;
    end;
    pp1 := pp;
    pp := pp^.next;
  end;
end;

{
procedure addEx(i : integer);
var
  ind : integer;
begin
  ind := find_el(i);
  if ind = 0 then
    add(i)
  else
    del(ind);
end;

procedure middel_and_del;
var
```

```
  pp : Tr;
  i, max, i_max, ind : integer;
begin
  pp := begptr;
  max := 0;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if (max < pp^.i) then
    begin
      max := pp^.i;
      i_max := ind;
    end;
    pp := pp^.next;
  end;
  del(i_max-1);
  for i:=i_max to i_max+5 do
    del(i);
end;
}

procedure sort;
var
  st:string;
  i,k,count:integer;
  pp,p,p1,p2: Tr;
  tmpBeg : Tr;
  find : boolean;
begin

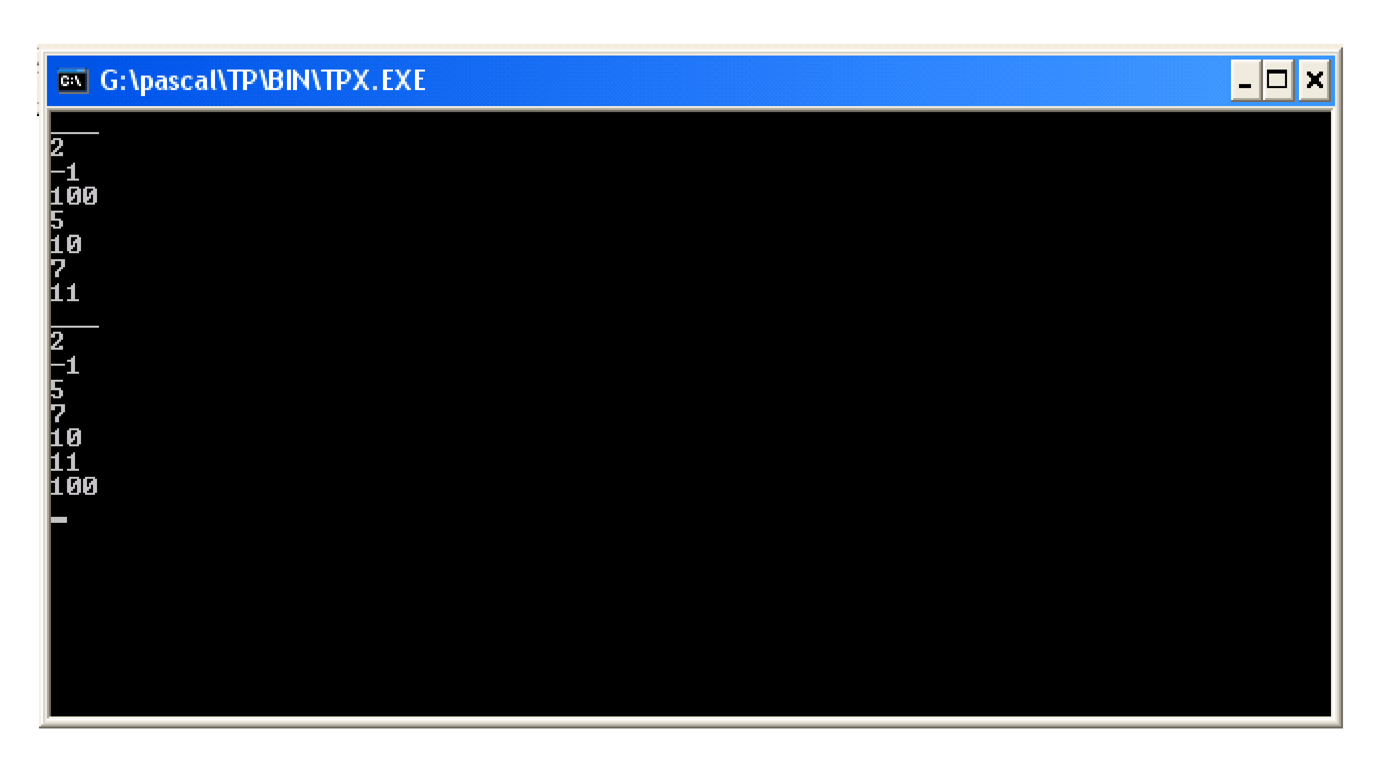
  find := false;
  pp := begptr;
  while pp <> nil do
  begin
    if pp^.i < 0 then
```

```
  begin
    find := true;
    break;
  end;
  pp := pp^.next;
end;
if not find then exit;
tmpBeg := begptr;
begptr := pp;        {after sorting we will delete this dummy element}
p:=begptr^.next;
count:=0;
while (p<>nil) do
begin
  inc(count);
  p:=p^.next;
end;

for i:=count downto 2 do
begin
  k:=1;
  p:=begptr;
  while (k<i) do
  begin
    p1:=p^.next;
    p2:=p1^.next;
    if (p1^.i > p2^.i) then
    begin
       p1^.next:=p2^.next;
      p2^.next:=p1;
      p^.next:=p2;
      if (p1^.next=nil) then
        endptr:=p1;
    end;
    p:=p^.next;
    inc(k);
  end;
```

```
  end;
  begptr := tmpBeg;
end;

begin
  clrscr;
  add(2,1);
  add(-1,2);
  add(10,3);
  add(5,3);
  add(7,5);
  add(11,6);
  add(100,3);
  writeln('___');
  show;
  sort;
  writeln('___');
  show;
end.
```



المثال الرابع

القائمة معطاة اوجد اكبر عنصر في القائمة ومن ثم مباشرة اعمل حذف لخمسة عناصر بعد هذا العنصر الاكبر .

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;


procedure add( i : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
    pp1 := endptr;
```

```
{   pp1 := find(i);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else}
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;
```

```
  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
```

```
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{       del2(pp1);}
  end;
end;

procedure middel_and_del;
var
  pp : Tr;
  i, max, i_max, ind : integer;
begin
  pp := begptr;
  max := 0;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if (max < pp^.i) then
    begin
      max := pp^.i;
      i_max := ind;
    end;
    pp := pp^.next;
  end;

           write('max:', max);
```

```
  for i:=i_max to i_max+5 do
    del(i);
end;

begin
  clrscr;
  add(2);
  add(10);
  add(5);
  add(1);
  add(7);
  add(11);
  add(2);
  add(3);
  add(1);
  add(6);
  add(4);
  add(1);
  add(3);
  add(9);
  writeln;
  show;
  middel_and_del;
  writeln;
  show;
end.
```

في البداية اوجدنا العنصر الاكبر كما في الجاريتم

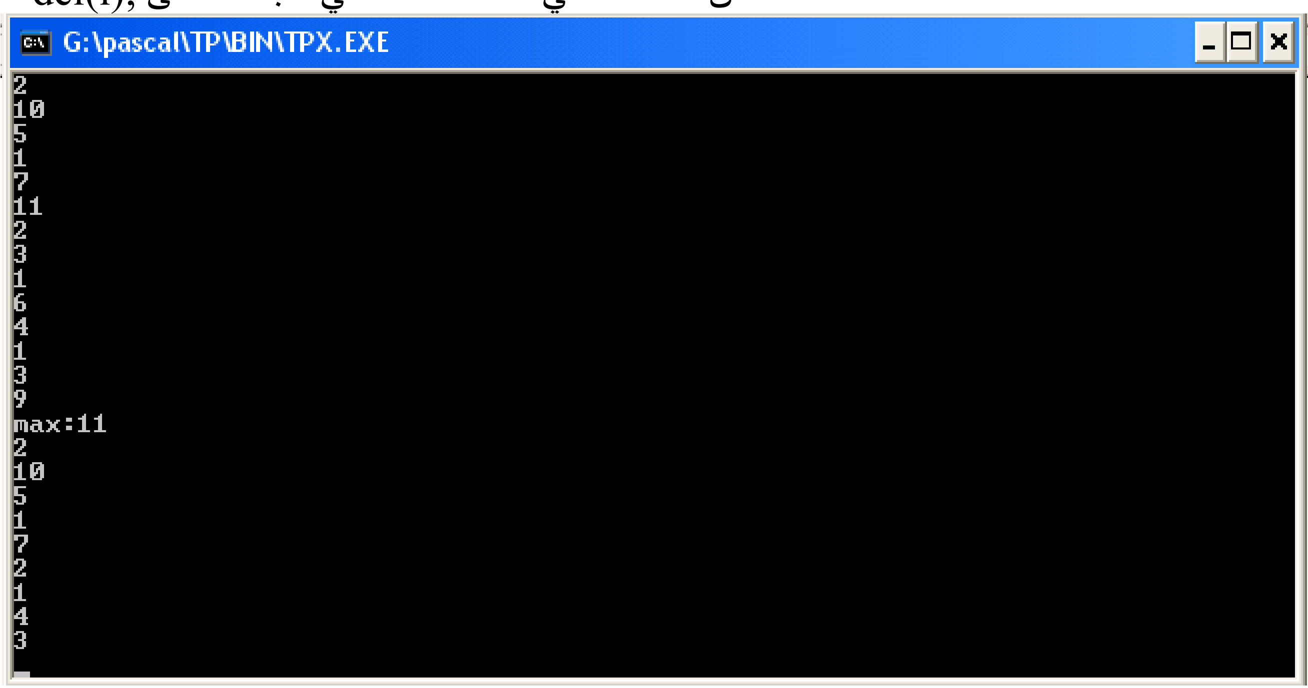
```
 if (max < pp^.i) then
    begin
      max := pp^.i;  اوجدنا هنا العنصر الاكبر
      i_max := ind;   هنااوجدناال position حق العنصر الاكبر
    end;
    pp := pp^.next;   من اجل ان يشير الى العنصر التالي
  end;
```

اظهار العنصر الاكبر

```
write('max:', max);
```

ومن ثم عملنا الحلقة تبدا من مكان العنصر الاكبر الى
5+ position مكان العنصر الاكبر

```
for i:=i_max to i_max+5 do
   del(i);
```
من هنا نستدعي دالة الحذف التي كتبناها اعلى



المثال الخامس

القائمة معطاة اوجدالعنصر الاكبر ومن ثم احذفة؟
البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
```

```
  begptr, endptr, pp : Tr;

procedure add( i : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
    pp1 := endptr;
{    pp1 := find(i);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else}
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
```

```
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
```

```
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{     del2(pp1);}
  end;
end;

procedure middel_and_del;
var
  pp : Tr;
```

```
  i, max, i_max, ind : integer;
begin
  pp := begptr;
  max := 0;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if (max < pp^.i) then
    begin
      max := pp^.i;
      i_max := ind;
    end;
    pp := pp^.next;
  end;

  writeln('max:=',max);
  del(i_max);

end;

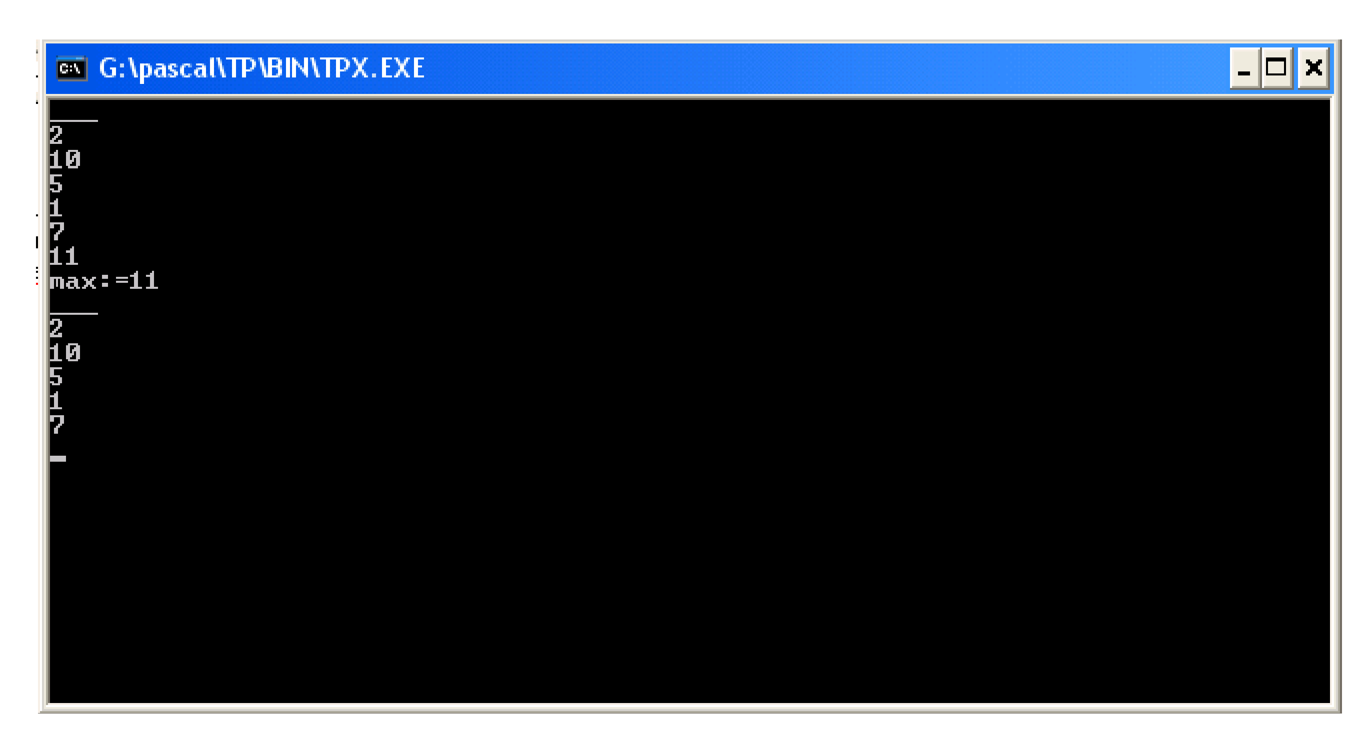
begin
  clrscr;
  add(2);
  add(10);
  add(5);
  add(1);
  add(7);
  add(11);
  writeln('___');
  show;
  middel_and_del;
  writeln('___');
  show;
end.
```

في البداية اوجدنا العنصر الاكبر كما في الجاريتم

```
 if (max < pp^.i) then
    begin
      max := pp^.i;  اوجدنا هنا العنصر الاكبر
      i_max := ind;   هنااوجدنال position حق العنصر الاكبر
    end;
    pp := pp^.next;   من اجل ان يشير الى العنصر التالي
  end;
```

اظهار العنصر الاكبر

```
write('max:', max);
del(i_max);ومن ثم نحذف ال position حق العنصر الاكبر
```



المثال السادس

القائمة معطاة اوجد العنصر ومن ثم احذفة من القائمة؟

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;

procedure add( i : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
    pp1 := endptr;
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
```

```
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
```

```
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{       del2(pp1);}
```

```
  end;
end;

begin
  clrscr;
  add(2);
  add(10);
  add(5);
  add(1);
  add(7);
  add(11);
  writeln('___');
  show;
  del(1);
  writeln('delete position number 1');
  show;
  del(3);
  writeln('delete position number 3');
  show;
  del(4);
  writeln('delete position number 4');
  show;
end.
```

```
G:\pascal\TP\BIN\TPX.EXE
2
10
5
1
7
11
delete position number 1
10
5
1
7
11
delete position number 3
10
5
7
11
delete position number 4
10
5
7
```

المثال السادس

القائمة معطاة اعمل على ترتيب العناصر في القائمة ومن ثم اعمل اضافة العنصر الذي تريدة في المكان الذي تريدة واحذف العنصر الذي تريدة من مكانة .

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;


function find_place(pl : integer) : Tr;
var
```

```
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
{   pp1 := endptr;{
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
```

```
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
```

```
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
```

```
      else
      begin
        pp2 := prev(pp1);
        pp1 := pp1^.next;
        dispose(pp2^.next);
        pp2^.next := pp1;
      end;
    exit;
  end;
{     del2(pp1);}
 end;
end;

procedure sort;
var
 st:string;
 x_beg, x_h, x_left, x_left_o:integer;
 i,k,count:integer;
 pp,p,p1,p2: Tr;
begin

 new(pp);
 pp^.next := begptr;  {for sorting needed first dummy element }
 begptr := pp;        {after sorting we will delete this dummy element}

 p:=begptr^.next;
 count:=0;
 while (p<>nil) do
 begin
  inc(count);
  p:=p^.next;
 end;

 for i:=count downto 2 do
 begin
  k:=1;
```

```
    p:=begptr;
    while (k<i) do
    begin
      p1:=p^.next;
      p2:=p1^.next;
      if (p1^.i > p2^.i) then
      begin
         p1^.next:=p2^.next;
        p2^.next:=p1;
        p^.next:=p2;
        if (p1^.next=nil) then
          endptr:=p1;
      end;
      p:=p^.next;
      inc(k);
    end;
  end;

  pp := begptr;
  begptr := begptr^.next;   {delete dummy element}
  dispose(pp);
end;
begin
  clrscr;
  add(2,1);
  add(10,2);
  add(5,3);
  add(1,4);
  add(7,5);
  writeln('___');
  show;
  sort;
  writeln('_sort element_');
  show;
  del(3);
  writeln('__delete position number 3');
```

```
  show;
  add(12,4);
writeln('__add in 4 position 12');
  show;
end.
```

```
G:\pascal\TP\BIN\TPX.EXE
____
2
10
5
1
7
_sort element_
1
2
5
7
10
__delete position number 3
1
2
7
10
__add in  position number 4 al3adad  12
1
2
7
12
10
```

المثال السابع

القائمة معطاة اوجد اكبر عدد في القائمة واصغرعدد في القائمة ومن ثم حذف اكبر عدد واصغر عدد والاعداد التي تقع بين اكبر واصغر عدد في القائمة .

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
    i : integer;
    next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;
```

```
function find_place(pl : integer) : Tr;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
{   pp1 := endptr;{
    pp1 := find(i); }
```

```
    pp1 := find_place(place);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;
```

```
  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
```

```
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{      del2(pp1);}
  end;
end;
procedure min_max;
var
  pp : Tr;
  i, max, i_max, ind : integer;
  min, i_min : integer;
  b, e : integer;
begin
  pp := begptr;
  max := 0;
  min := begptr^.i;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if (max < pp^.i) then
    begin
      max := pp^.i;
      i_max := ind;
    end;
    if (min > pp^.i) then
    begin
```

```
      min := pp^.i;
      i_min := ind;
    end;
    pp := pp^.next;
  end;

  if i_max > i_min then
  begin
    b := i_min;
    e := i_max;
  end
  else
  begin
    b := i_max;
    e := i_min;
  end;
    writeln('___');
  writeln('max:=',max);
  writeln('min:=',min);
 for i:=e downto b do
    del(i);
end;

begin
  clrscr;
  add(2,1);
  add(10,2);
  add(5,3);
  add(1,4);
  add(7,5);
  add(11,6);

  add(100,2);
```

```
  show;
  min_max;
  writeln('___');
  show;
end.
```

```
if (max < pp^.i) then هنا نوجد اكبر عدد
    begin
      max := pp^.i; اكبر عدد
      i_max := ind; ال position حق اكبر عدد
    end;
    if (min > pp^.i) then هنا نوجد اصغر عدد
    begin
      min := pp^.i; اصغر عدد
      i_min := ind; هنا نوجد position حق اصغر عدد
    end;
    pp := pp^.next; يشير الى العنصر القادم
```

هنا نوجد المكان الذي يقع بين اكبر واصغر عدد في القائمة

```
  if i_max > i_min then اذا كان position اكبر عنصر اكبر من positionاصغر عنصر
  begin
    b := i_min;    positionاصغر عنصر
    e := i_max;    positionاكبر عنصر
  end
  else
  begin
    b := i_max;positionاكبر عنصر
    e := i_min;positionاصغر عنصر
  end;

  for i:=e downto b do
```

del(i);   نحذف i التي هي اكبر واصغر عدد والاعداد التي بينهما

```
G:\pascal\TP\BIN\TPX.EXE
2
100
10
5
1
7
11
___
max:=100
min:=1
___
2
7
11
```

المثال الثامن

القائمة معطاة اوجد اكبر عنصر فيها ومن ثم اعمل على تقسيم القائمة الى قائمتين
قائمة تبدا من اول عنصر الى ما قبل اكبر عنصر وقائمة تبدا من اكبر عنصر الى اخر عنصر.

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
    i : integer;
    next : Tr;
  end;
var
  i : integer;
```

```
  begptr, endptr, pp : Tr;
  list1_begptr, list1_endptr : Tr;
  list2_begptr, list2_endptr : Tr;


function find_place(pl : integer) : Tr;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
```

```
  else
  begin
{    pp1 := endptr;{
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
```

```
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
```

```
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{      del2(pp1);}
  end;
end;

procedure max_split;
var
  pp : Tr;
  i, max, i_max, ind : integer;
  min, i_min : integer;
  pp_prev, p_prev, p_max : Tr;
  b, e : integer;
begin
  pp := begptr;
  max := 0;
  min := begptr^.i;
  ind := 0;
  p_prev := begptr;
  p_max := nil;
  while pp <> nil do
  begin
```

```
  inc(ind);
  if (max < pp^.i) then
  begin
    max := pp^.i;
    i_max := ind;
    p_prev := pp_prev;
    p_max := pp;
  end;
  pp_prev := pp;
  pp := pp^.next;
end;
writeln;
writeln('MAX:', p_max^.i);

list1_begptr := begptr;
list1_endptr := p_prev;
list1_endptr^.next := nil;


list2_begptr := p_max;
list2_endptr := endptr;
list2_endptr^.next := nil;
writeln('__list1__');
pp := list1_begptr;
while pp <> nil do
begin
  writeln(pp^.i);
  pp := pp^.next;
end;
writeln('__list2__');
pp := list2_begptr;
while pp <> nil do
begin
  writeln(pp^.i);
  pp := pp^.next;
end;
```

```
end;

begin
  clrscr;
  add(2,1);
  add(10,2);
  add(5,3);
  add(1,4);
  add(7,5);
  add(11,6);
  add(100,4);
  show;
  max_split;
  writeln('___');

end.
```

```
 if (max < pp^.i) thenنوجد اكبر عنصر
    begin
      max := pp^.i; اكبر عنصر
      i_max := ind;  مكان  اكبر عنصر
p_prev العنصر الذي قبل اكبر عنصر
pp_prev العنصر الاخير في القائمة
العنصر الاخير في القائمة  = العنصر الذي قبل اكبر عنصر
    p_prev := pp_prev;
      p_max := pp; اكبر عنصر عملنا لة بريسفايفانيا
    end;
    pp_prev := pp;العنصر الاخير في القائمة
    pp := pp^.next; يوشر الى العنصر القادم
  end;
```

```
  writeln;
  writeln('MAX:', p_max^.i); اظهار اكبر عنصر

  list1_begptr := begptr; في القائمة الاولى نضع فيها من بداي القائمة الرئيسية
  list1_endptr := p_prev;  نضع في القائمة الاولى الى ما قبل اكبر عنصر
  list1_endptr^.next := nil; نعمل عملية التصفير  نجعل النهاية تشير الى الصفر


  list2_begptr := p_max; في القائمة الثانية نضع فيها في البداية اكبر عنصر حيثما توقفنا
  list2_endptr := endptr;حتى النهاية الى اخر عنصر
  list2_endptr^.next := nil;نعمل عملية التصفير  نجعل النهاية تشير الى الصفر
  writeln('__list1__');
  pp := list1_begptr;      pp القائمة الاولى  نرميها الى المتغير من نفس النوع
  while pp <> nil do    نفحص فيما اذا كانت القائمة فاضية او لا
  begin
    writeln(pp^.i);        نعمل على اظهار القائمة الاولى
    pp := pp^.next; من اجل ان يشير الى العناصر التالية في القائمة الاولى ويعمل على اظهارها
كاملة
  end;
  writeln('__list2__');
  pp := list2_begptr;      pp القائمة الثانية الى المتغير من نفس النوع
  while pp <> nil do    نفحص فيما اذا كانت القائمة فاضية او لا
  begin
    writeln(pp^.i);نعمل على اظهار القائمة الثانية

    pp := pp^.next; من اجل ان يشير الى العناصر التالية في القائمة الثانية ويعمل على اظهارها
كاملة
  end;
```

```
G:\pascal\TP\BIN\TPX.EXE
2
10
5
100
1
7
11

MAX:100
__list1__
2
10
5
__list2__
100
1
7
11
____
```

المثال التاسع

القائمة معطاة احذف من القائمة الاعداد التي تقبل القسمة على ثلاثة وسبعة وضع في مكانها

العدد 777.

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;
function find_place(pl : integer) : Tr;
var
```

```
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
{   pp1 := endptr;{
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
```

```
    pp^.next := begptr;
    begptr := pp;
  end
  else
  begin
    if pp1^.next = nil then
      endptr := pp;
    pp^.next := pp1^.next;
    pp1^.next := pp;
  end;
 end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
```

```
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
          pp2^.next := nil;
          dispose(endptr);
          endptr := pp2;
        end
```

```
        else
        begin
          pp2 := prev(pp1);
          pp1 := pp1^.next;
          dispose(pp2^.next);
          pp2^.next := pp1;
        end;
      exit;
    end;
{     del2(pp1);}
  end;
end;


procedure edit;
var
  pp : Tr;
  i, count : integer;
begin
  pp := begptr;
  count := 0;
  while pp <> nil do
  begin
    inc(count);
    pp := pp^.next;
  end;
  writeln('count:', count);

  for i:=1 to count do
  begin
    pp := find_place(i);
    if (pp^.i mod 3 = 0) or (pp^.i mod 7 = 0) then
    begin
      del(i-1);
      add(777, i-1);
    end;
```

```
  end;

  writeln('__');
  pp := begptr;
  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;

end;

begin
  clrscr;
  add(7,1);
  add(10,2);
  add(3,3);
  add(1,4);
  add(21,5);
  add(22,6);
  add(33,7);
  add(42,8);
  add(88,9);
  writeln('___');
  show;
  edit;
  writeln('___');

end.
```

```
pp := begptr;  بداية القائمة
  count := 0;  من اجل نوجد كمية الاعداد

  while pp <> nil do نفحص فيما اذا كانت القائمة فاضية
  begin
    inc(count); كمية العناصر
    pp := pp^.next; من اجل يشير الى العناصر القادمة
  end;
  writeln('count:', count); اظهار الكمية

  for i:=1 to count do  نعمل الحلقة على حسب الكمية
  begin
    pp := find_place(i);  اوجدنا مكان العنصر
   الاعداد التي تقبل القسمة على 3 او على 7
    if (pp^.i mod 3 = 0) or (pp^.i mod 7 = 0) then
    begin
      del(i-1);  نحذف مكان العنصر
      add(777, i-1);  ونضع في مكان العنصر العدد 777
    end;
  end;

  writeln('__');
  pp := begptr;  من بداية القائمة
  while pp <> nil do  نفحص فيما اذا كانت القائمة فاضية
  begin
    writeln(pp^.i); نعمل على اظهار النتيجة
    pp := pp^.next; من اجل ان يشير الى العناصر القادمة وتظهر كاملة
  end;
```

```
G:\pascal\TP\BIN\TPX.EXE
____
7
10
3
1
21
22
33
42
88
count:9
__
777
10
777
1
777
22
777
777
88
____
_
```

المثال العاشر

القائمة معطاة قم بتقسيم القائمة الى ثلاثة اقسام القسم الاول من البداية الى ما قبل اصغر عدد والقسم الثاني يبدا من اصغر عدد الى ما قبل اكبر عدد والقسم الثالث يبدا من اكبر عدد الى النهاية.

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
     i : integer;
     next : Tr;
  end;
var
  i : integer;
  begptr, endptr, pp : Tr;
  list1_begptr, list1_endptr : Tr;
  list2_begptr, list2_endptr : Tr;
```

```
  list3_begptr, list3_endptr : Tr;

function find_place(pl : integer) : Tr;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
  begin
{    pp1 := endptr;{
```

```
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
  pp := begptr;
  pp1 := nil;
```

```
  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
        begin
          pp2 := prev(pp1);
```

```
        pp2^.next := nil;
        dispose(endptr);
        endptr := pp2;
      end
      else
      begin
        pp2 := prev(pp1);
        pp1 := pp1^.next;
        dispose(pp2^.next);
        pp2^.next := pp1;
      end;
    exit;
  end;
{     del2(pp1);}
 end;
end;

procedure max_split;
var
  pp : Tr;
  i, max, i_max, ind : integer;
  min, i_min : integer;
  pp_prev, p_prev, p_max, p_min, p_prev2 : Tr;
  b, e : integer;
begin
  pp := begptr;
  max := 0;
  min := begptr^.i;
  ind := 0;
  p_prev := begptr;
  p_max := nil;
  while pp <> nil do
  begin
    inc(ind);
    if (max < pp^.i) then
    begin
```

```
    max := pp^.i;
    i_max := ind;
    p_prev := pp_prev;
    p_max := pp;
  end;

  if (min > pp^.i) then
  begin
    min := pp^.i;
    i_min := ind;
    p_prev2 := pp_prev;
    p_min := pp;
  end;

  pp_prev := pp;
  pp := pp^.next;
 end;
{  writeln('MAX:', p_max^.i);}

 list1_begptr := begptr;
 list1_endptr := p_prev2;
 list1_endptr^.next := nil;


 list2_begptr := p_min;
 list2_endptr := p_prev;
 list2_endptr^.next := nil;

 list3_begptr := p_max;
 list3_endptr := endptr;
 list3_endptr^.next := nil;

 writeln('__list1__');
 pp := list1_begptr;
 while pp <> nil do
 begin
```

```
    writeln(pp^.i);
    pp := pp^.next;
  end;

  writeln('__list2__');
  pp := list2_begptr;
  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;

  writeln('__list3__');
  pp := list3_begptr;
  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;
begin
  clrscr;
  add(2,1);
  add(10,2);
  add(1,3);
  add(5,4);
  add(7,5);
  add(11,6);
  add(100,5);
  writeln('___');
  show;
  writeln('___');
  writeln('altaqsim:');
  max_split;

end.
```

شرحها تقريبا نفس البرنامج رقم8

```
G:\pascal\TP\BIN\TPX.EXE
___
2
10
1
5
100
7
11
___
altaqsim:
__list1__
2
10
__list2__
1
5
__list3__
100
7
11
```

المثال الحادي عشر

قائمتين معطاة اعمل على دمج القائمتين مع بعض.

البرنامج

```
uses crt;
type
  Tr = ^r;
  r = record
    i : integer;
    next : Tr;
  end;
var
  i : integer;
```

```
  begptr, endptr, pp : Tr;
  list1_begptr, list1_endptr : Tr;
  list2_begptr, list2_endptr : Tr;

function find_place(pl : integer) : Tr;
var
  pp, pp1 : Tr;
  ind : integer;
begin
  pp1 := nil;
  pp := begptr;
  ind := 0;
  while pp <> nil do
  begin
    inc(ind);
    if pl = ind then
      break;
    pp1 := pp;
    pp := pp^.next;
  end;
  find_place := pp1;
end;

procedure add( i : integer; place : integer );
var
  pp, pp1 : Tr;
begin
  new (pp);
  pp^.i := i;
  pp^.next := nil;
  if endptr = nil then
  begin
    endptr := pp;
    begptr := pp;
  end
  else
```

```
  begin
{    pp1 := endptr;{
    pp1 := find(i); }
    pp1 := find_place(place);
    if pp1 = nil then
    begin
      pp^.next := begptr;
      begptr := pp;
    end
    else
    begin
      if pp1^.next = nil then
        endptr := pp;
      pp^.next := pp1^.next;
      pp1^.next := pp;
    end;
  end;
end;

procedure show;
var
  pp : Tr;
begin
  pp := begptr;

  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;
end;

function prev(p : Tr) : Tr;
var
  pp, pp1 : Tr;
begin
```

```
  pp := begptr;
  pp1 := nil;

  while pp <> nil do
  begin
    if pp^.next = p then
    begin
      pp1 := pp;
      break;
    end;
    pp := pp^.next;
  end;
  prev := pp1;
end;

procedure del(ind : integer);
var
  pp, pp1, pp2 : Tr;
  i : integer;
begin
  pp := begptr;
  i := 0;
  while pp <> nil do
  begin
    i := i + 1;
    pp1 := pp;
    pp := pp^.next;
    if i = ind then
    begin
      if pp1 = begptr then
      begin
        begptr := begptr^.next;
        dispose(pp1);
      end
      else
        if pp1 = endptr then
```

```
      begin
        pp2 := prev(pp1);
        pp2^.next := nil;
        dispose(endptr);
        endptr := pp2;
      end
      else
      begin
        pp2 := prev(pp1);
        pp1 := pp1^.next;
        dispose(pp2^.next);
        pp2^.next := pp1;
      end;
    exit;
  end;
{     del2(pp1);}
 end;
end;

begin
  clrscr;
  add(2,1);
  add(10,2);
  add(1,3);
  add(5,4);
  list1_begptr := begptr;
  list1_endptr := endptr;
  begptr := nil;
  endptr := nil;

  add(7,5);
  add(11,6);

  add(100,5);
  list2_begptr := begptr;
  list2_endptr := endptr;
```

```
  writeln('__list1__');
  pp := list1_begptr;
  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;

  writeln('__list2__');
  pp := list2_begptr;
  while pp <> nil do
  begin
    writeln(pp^.i);
    pp := pp^.next;
  end;

  begptr := nil;
  endptr := nil;
  writeln;
  writeln('__ete7ad  list1+list2:__');
  begptr := list1_begptr;
  list1_endptr^.next := list2_begptr;
  endptr := list2_endptr;
  show;
  writeln('___');
end.
```

G:\pascal\TP\BIN\TPX.EXE

```
__list1__
2
10
1
5
__list2__
7
11
100

__ete7ad  list1+list2:__
2
10
1
5
7
11
100
____
```

Clrscr; تصفية الشاشة

position اضافةالعناصر الى القائمة الاولى حسب المكان

```
add(2,1);
add(10,2);
add(1,3);
add(5,4);
```

list1_begptr := begptr; بداية القائمة الاولى
list1_endptr := endptr; نهاية القائمة الثانية
begptr := nil; البداية نصفر
endptr := nil;النهاية نصفر

position اضافةالعناصر الى القائمة الثانية حسب المكان

```
add(7,5);
add(11,6);

add(100,5);
```

list2_begptr := begptr; بداية القائمة الثانية
list2_endptr := endptr; نهاية القائمة الثانية

هنا نعمل على اظهار عناصر القائمة الاولى

```
  writeln('__list1__');
  pp := list1_begptr; من البداية
  while pp <> nil do  نفحص فيما اذا كانت القائمة فاضية ام لا
  begin
    writeln(pp^.i); هنا نعمل على اظهار عناصر القائمة الاولى
    pp := pp^.next; اظهار العناصر المتبقية في القائمة الاولى
  end;
```

هنا نعمل على اظهار عناصر القائمة الثانية

```
  writeln('__list2__');
  pp := list2_begptr; من البداية
  while pp <> nil do نفحص فيما اذا كانت القائمة فاضية ام لا
  begin
    writeln(pp^.i);هنا نعمل على اظهار عناصر القائمة الثانية
    pp := pp^.next;اظهار العناصر المتبقية في القائمة الثانية
  end;

  begptr := nil; نصفر  البداية
  endptr := nil;نصفر النهاية
  writeln;
  writeln('__ete7ad  list1+list2:__');
```

في القائمة الرئيسيةالجديدة في بدايتها نضع بداية القائمة الاولى

```
  begptr := list1_begptr;
```

ونهاية القائمة الاولى تظل توشر الى العناصر القادمة من بداية القائمة الثانية

```
  list1_endptr^.next := list2_begptr;
```

في نهاية القائمة الجديدة ندخل نهاية القائمة الثانية

```
  endptr := list2_endptr;
  show;  ومن ثم نستدعي الدالة لاظهار النتيجة للدمج
```

وفي الاخير دعواتكم عبد الماجد الخليدي
روسيا الاتحادية ـ روستوف نادانو